

Atari 810 Disk Drive
Controller Program Report

18 Nov 80
Dave Strauss

Development of the Atari 810 disk controller program was begun in the later part of 1978 by Bob Warne using a KIM development system. The various routines started off as test software to check the operation of newly designed disk drive hardware. Eventually, the hand-assembled software evolved and got into the Rev. B program that is found in the production ROM.

The controller program works quite well. No serious bugs have been observed. Considering the way it was developed, it is amazing that Bob Warne could have produced 2K of nearly bug free code by piecing together hand-assembled routines and entering the hex code via the KIM keyboard. Documentation and maintainability are the big problems with such development procedures however. It is my endeavor with this report to bring these necessities somewhat up to snuff.

To this end, I got a copy of the disassembled source file (corresponding to the current Rev. D) as generated by Ian Sheppard Aug. 1979 from the controller program hex code object. I studied it, drew flow charts and added comments (and even removed over 24,000 ASCII blank spaces from it!!!). Now, with the comments, notes, symbolic references, labels for constants and flags, and a cross-reference added, the source file should be much easier to understand and maintain.

In addition to improving the state of the source for the existing controller code, I took it upon myself to modify an admittedly bug-free program (my unofficial Rev. E). This was done so that the structure would become more straightforward, and the operation a little more flexible. It may become evident that I've made the code a little too flexible, considering the ongoing concerns for disk software protection.

The changes I have made to constitute the present unofficial Rev. E are basically:

1. an ordering of all the routines into a more logical sequence minimizing excessive JMP's.
2. the addition of a new "user-command" table entry which will execute code previously downloaded into the 64 available bytes of the controller RAM.
3. the change of addressing modes used by the format

routine to enable the entry associated with command to substitute its own sector number and a "bad bunny format" (bad sectors created with inaccessible sector numbers), and

4. the debussing of the two other routines added by Rich Grass & George Simcock for reading the extended status of the controller (Read from Address) and Download for entering in-house diagnostic software (both routines first appear in Rev. D).

One notable thing about the Rev. B and Rev. C versions (the only versions in use) is the fact that the stack pointer isn't initialized at power-up but is used by a JSR/RTS before the stack finally gets init'ed at the warm-start entry point. This hasn't seemed to cause any problems, evidently because the stack pointer register powers up in a benign state (=FF). It should be noted however that if the SP register powered up with a value less than \$80, the stack would map into non-existent RAM causing that first JSR to lose the return address. Stack pointer initialization is first found in the Rev. D version of the disk controller program.

A summary of the disk controller Rev. changes:

Rev. A to Rev. B c. Nov 78-Apr 79

N/A; no data on Rev. A

Rev. B to Rev. C Rev. B burned into ROM about Dec 79
Rev. C made in EPROM only about Apr 80

1. The sector table used by Format was changed
from 18,7,14,3,16,17,6,13,2,9,16,5,12,1,8,15,4,11
to 18,1,3,5,7,9,11,13,15,17,2,4,6,8,10,12,14,16
2. The algorithm (a table wasn't used) to verify sectors after Format changed effectively
from 17,13,9,5,1,15,11,7,3,16,12,8,4,18,14,10,6,2
to 1,5,9,13,17,4,8,12,16,3,7,11,15,2,6,10,14,18
3. No other changes noted.

Rev. C to Rev. D Rev. D made about Jul-Aug 80

1. In Rev D., the stack is initialized immediately after power-up to \$FF.
2. The routine to sense the diskette write protect status is moved from the power-up initialization code (in Rev. C) down to the warm start entry point (in Rev. D). Code that initializes the Status command time out value (LSB & MSB) is likewise moved.

3. A download routine is inserted in the space between the last JMP from Initialization to Initialization, Part II, and the sector table (which needs to terminate on a page boundary for the Format logic to work).
4. A .BYTE instruction following a JMP in the read routine and commented as "Junk", is removed.
5. The byte used 128 times in the data field for each sector written during a Format is changed from 00 (inverted to \$FF on media), to \$E5 (inverted to \$1A).
6. The RTS following the JMP instruction (which can never get control) in the routine to update the STATUS register is removed.
7. The Command Interpreter routine is modified to use a table of legal commands used to find the index into LSB and MSB address tables that form the indirect jump address (Rev. D) instead of an awkward computed address algorithm (with several special cases) into a table of JMP instructions, several of which JMP to invalid command handling (Rev. C). The legal commands are expanded from 6 to 8 to include Read from Address and Download.
8. A Read From Address routine is added in the remaining ROM address space. However, the routine fails to send ACK over serial bus upon receipt of the command and it doesn't set the command complete code (CMPLT) in SDATA prior to data field transmission, so it is inconceivable that it would work.

Rev. D to Rev. E

Rev. E made Oct 80

1. All routines are re-ordered so that the main monitor loop follows init, various two-part routines are re-united, and the logic in Format is changed to allow a location independent sector number table.
2. All working RAM registers are consolidated to start from \$1C0 up to \$1DB inclusive, increasing available stack size from 10 to 17 nested subroutine calls, and making 64 contiguous bytes of RAM available for downloaded software.
3. The Format routine uses an address independent 18-sector table rather than the address dependant 17-sector table previously used. Sector verify is performed on Format by my own read sector driver which waits for Ready & Not Busy from the 1771 status register before reading instead of Force Interrupting the 1771 each time.
4. The error retry count (currently set to 4) for sector reads and writes is now initialized in RAM at power-up only, instead of being built into the ROM code so that a user may alter that count for his own needs.
5. The Status command now waits for the command frame line to go lo before sending the ACK code.
6. Read from Address and Download now follow SIO protocol.
7. Record Not Found error status from the 1771 after the third retry of a read or write now causes a seek to track 0, then a reseek to the track with the error, instead of merely stepping out 43 times and then reseeking.
8. Time delay after head home is provided by existing timer

relative rather than the in-line code previously used.

- + Change in motor shutdown time from 7 seconds to 3 seconds.
- + Doesn't turn on motor as part of controller initialization.
- + RAM test?

2430

Rich Gregg

Atari 810 Disk Drive Controller Program Report

18 Nov 80
Dave Stauss

Development of the Atari 810 disk controller program was begun in the later part of 1978 by Bob Warne using a KIM development system. The various routines started off as test software to check the operation of newly designed disk drive hardware. Eventually, the hand-assembled software evolved and into the Rev. B program that is found in the production ROM.

The controller program works quite well. No serious bugs have been observed. Considering the way it was developed, it is amazing that Bob Warne could have produced 2k of nearly bug free code by piecing together hand-assembled routines and entering the hex code via the KIM keyboard. Documentation and maintainability are the big problems with such development procedures however. It is my endeavor with this report to bring these necessities somewhat up to snuff.

To this end, I got a copy of the disassembled source file (corresponding to the current Rev. D) as generated by Ian Sheppard Aug. 1979 from the controller program hex code object. I studied it, drew flow charts and added comments (and even removed over 24,000 ASCII blank spaces from it!!!). Now, with the comments, notes, symbolic references, labels for constants and flags, and a cross-reference added, the source file should be much easier to understand and maintain.

In addition to improving the state of the source for the existing controller code, I took it upon myself to modify an admittedly bug-free program (my unofficial Rev. E). This was done so that the structure would become more straightforward, and the operation a little more flexible. It may become evident that I've made the code a little too flexible, considering the ongoing concerns for disk software protection.

The changes I have made to constitute the present unofficial Rev. E are basically:

1. an ordering of all the routines into a more logical sequence minimizing excessive JMP's,
2. the addition of a new "user-command" table entry which will execute code previously downloaded into the 64 available bytes of the controller RAM,
3. the change of addressing modes used by the format

routine to enable the above mentioned user command to substitute its own sector number table for "funny formats" (bad sectors created with unaccessible sector numbers), and

2. the debugging of the two other routines used by Rich Grage & George Simcoe for reading the extended status of the controller (Read from Address) and Download for interne in-house diagnostic software (both routines first appear in Rev. D).

One notable thing about the Rev. B and Rev. C versions (the only versions in used is the fact that the stack pointer isn't initialized at power up but is used by a JSR RTS before the stack finally gets init'ed at the warm-start entry point. This hasn't seemed to cause any problems, apparently because the stack pointer register comes up in a random state (FFF). It should be noted however that if the SP register showed up with a value less than \$20, the stack would map into non-existent RAM causing that first JSR to lose the return address. Stack pointer initialization is first found in the Rev. D version of the disk controller program.

A summary of the disk controller Rev. changes:

Rev. A to Rev. B 1. Nov 72-Apr 73

Rev. A, no date or Rev. A

Rev. B to Rev. C Rev. B started into ROM about Dec 72
Rev. C came in EPROM only about Apr 73

1. The sector table used by Format was changed:
from 10,7,14,3,10,17,6,10,2,8,10,3,12,1,8,17,4,11
to 10,1,2,5,7,9,11,13,15,17,2,4,6,8,10,12,14,16
2. The algorithm (a table wasn't used) to verify sectors after Format changed effectively:
from 17,13,3,5,1,15,11,7,3,16,12,3,4,13,14,10,2
to 1,5,9,16,17,4,8,12,16,6,7,11,15,1,5,10,14,18
3. No other changes noted.

Rev. C to Rev. D Rev. D made about Dec-Apr 73

1. In Rev D., the stack is initialized immediately after power-up to FFF.
2. The routine to read the cassette drive sector status is moved from the power-up initialization code in Rev. C down to the warm-start entry point (in Rev. D., code that initializes the Status command time out value 110 & 100 is likewise moved).

3. A download routine is inserted in the space between the last JMP from Initialization to Initialization, Part II, and the sector table (which needs to terminate on a page boundary for the Format logic to work).
4. A .BYTE instruction following a JMP in the read routine and commented as "Junk", is removed.
5. The byte used 128 times in the data field for each sector written during a Format is changed from 00 (inverted to \$FF on media), to \$E5 (inverted to \$1A).] why?
6. The RTS following the JMP instruction (which can never get control) in the routine to update the STAT0 register is removed.
7. The Command Interpreter routine is modified to use a table of legal commands used to find the index into LSB and MSB address tables that form the indirect jump address (Rev. D) instead of an awkward computed address algorithm (with several special cases) into a table of JMP instructions, several of which JMP to invalid command handlers (Rev. C). The legal commands are expanded from 6 to 8 to include Read from Address and Download.
8. A Read From Address routine is added in the remaining ROM address space. However, the routine fails to send ACK over serial bus upon receipt of the command and it doesn't set the command complete code (CMPLT) in SDATA prior to data field transmission, so it is inconceivable that it would work.

Rev. D to Rev. E

Rev. E made Oct 80

1. All routines are re-ordered so that the main monitor loop follows init, various two-part routines are re-united, and the logic in Format is changed to allow a location independent sector number table.
2. All working RAM registers are consolidated to start from \$1C0 up to \$1DB inclusive, increasing available stack size from 10 to 17 nested subroutine calls, and making 64 contiguous bytes of RAM available for downloaded software.
3. The Format routine uses an address independent 18-sector table rather than the address dependant 17-sector table previously used. Sector verify is performed on Format by my own read sector driver which waits for Ready & Not Busy from the 1771 status register before reading instead of Force Interrupting the 1771 each time.
4. The error retry count (currently set to 4) for sector reads and writes is now initialized in RAM at power-up only, instead of being built into the ROM code so that a user may alter that count for his own needs.
5. The Status command now waits for the command frame line to go lo before sending the ACK code.
6. Read from Address and Download now follow SIO protocol.
7. Record Not Found error status from the 1771 after the third retry of a read or write now causes a seek to track 0, then a reseek to the track with the error, instead of merely stepping out 43 times and then reseeking.]?
8. Time delay after head home is provided by existing timer

routine rather than the in-line code previously used.

